**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

Applicants: Parruck, et al.

Assignee: Azanda Network Devices

Title: "Multi-Service Segmentation and Reassembly Device That Maintains Reduced Number of Segmentation Contexts" (As Amended)

Serial No.: Unknown Filed: October 12, 2001

Examiner: Unknown Group Art Unit: Unknown

Atty. Doc. No.: AZA-003-5D/2001-P008

---

October 12, 2001

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

# PRELIMINARY AMENDMENT

Sir:

Before action on the merits, and before calculating the filing fee, please amend the above-identified application as follows.

IN THE DRAWINGS:

A minor error in Figure 52 is corrected. **A copy of Figure 52 with the correction indicated in red is attached.** In reviewing the correction, the Examiner's attention is direct to page 59, line 22 of the specification through page 60, line 21 and the corrections made to that text (see below). Review and approval of the correction by the Examiner is requested. No subject matter is added by the correction, nor is it the intent of Applicants or the undersigned that any subject matter be added.

IN THE TITLE:

Please change the title of the application to: "Multi-Service Segmentation and Reassembly Device That Maintains Reduced Number of Segmentation Contexts".

IN THE SPECIFICATION:

**Page 1, before the first sentence of the specification, please add the following header and paragraph:**

CROSS-REFERENCE TO RELATED APPLICATION

This application claims priority under 35 U.S.C. §120 from U.S. Patent Application Serial Number 09/851,565, filed May 8, 2001.

**The paragraph from page 14, line 31 to page 15, line 10 should be replaced with:**

Figure 4 is a simplified diagram of a router 100 in accordance with an embodiment of the present invention. Router 100 includes a plurality of line cards 101-104, a switch fabric 105 and a central processing unit (CPU) 106. The line cards 101-104 are coupled to switch fabric 105 by parallel buses 107-114. In the present example, each of parallel buses 107-114 is a 16-bit SPI-4, Phase II, LVDS parallel bus operating at 400 MHz at a double data rate (DDR). CPU 106 is coupled to line cards 101-104 by another parallel bus 131. In the present example, parallel bus 131 is a 32-bit PCI bus. In this example, each of the line cards can receive network communications in multiple formats. For example, line card 101 is coupled to a fiber optic cable 115 such that line card 101 can receive from cable 115 network communications at OC-192 rates in packets, ATM cells, and/or AAL5 cells. AAL5 cells are considered a type of ATM cell.

**Page 33, the first full paragraph, should be replaced with:**

In accordance with egress application type 11, segmentation block 203 does not perform segmenting per se but rather forwards the 64-byte chunk to memory manager block 204 via the per-port FIFO mechanism described above. Memory manager block 204 issues an enqueue command via enqueue command line 320, and stores the 64-byte chunk in payload memory 217. Per flow queue block 207 adds the BID to the per flow queue for flow #1. The same process flow occurs for the second and third switch cells associated with flow #1.

**Page 34, the second full paragraph, should be replaced with:**

Figure 32 illustrates the general flow of information out of egress MS-SAR 200 in one scenario when the linked lists for flow #1 and flow #2 are dequeued. As each BID is dequeued from a per flow queue, the associated 64-byte chunk is read from payload memory 217 and is supplied via 128-bit wide data bus 324 to reassembly block 205. As explained above in connection with the ingress mode, reassembly block 205 maintains one reassembly queue for each of the 64 logical output ports. If two flows share the same logical output port, then the entire linked list for one flow must be dequeued before the linked list for the next flow having the same logical output port is dequeued. In the example of Figure 8, flow #1 and flow #2 have different logical output ports. The dequeue command as received by memory manager block 204 from per flow queue block 207 contains the port ID (PID) of the one of the 64 logical output ports of line card 103.

**Page 36, the second full paragraph, should be replaced with:**

The AAL5 trailer also contains a sixteen-bit "length of payload" field that indicates the length of the payload of the packet being reassembled. Reassembly block 205 maintains a partial packet byte count value for the packet. As each 64-byte chunk of the packet is received from memory manager block 204, reassembly block 205 adds the number of bytes received to the partial packet byte count value. After the last 64-byte chunk for the packet has been received and the partial packet byte count value has been added to, reassembly block 205 compares the now complete packet byte count value with the "length of payload" value from the AAL5 trailer. The two should match.

**Page 59, the fourth full paragraph, should be replaced with:**

Figure 52 is a diagram of an external memory device 529 that is coupled to a control integrated circuit. External memory device 529 may, for example, be external memory 225 of Figure 10. External memory device 529 stores two types of information, information #1 and information #2, both of which must be accessed within a particular amount of time related to the rate of incoming information. Where the control integrated circuit is clocked by a clock signal, this particular amount of time can be referred to as

four clock periods. In the example to the left of Figure 52, each piece of information can be accessed in two clock periods. Both pieces of information are stored in the same external memory device requiring one to be accessed before the other. A total of four clock periods is therefore required to access both pieces of information.

**Page 60, the second full paragraph, should be replaced with:**

In accordance with one novel aspect, two external memories are used. Information #1 is stored in the first external memory 530 and information #2 is stored in second external memory 531. The two external memories are accessed at the same time in parallel. If external memories 530 and 531 are the same type of external memory 529, then these external memories have access times of two clock periods and both information #1 and information #2 are accessed within the required two clock periods. It is to be understood, of course, that the four and two in the example are used only to illustrate the technique of accessing memories in parallel to facilitate handling higher data throughput rates. The technique here is not limited to the numbers in this example. An example of information #1 is cell count information stored in the embodiment of Figure 10 in PFQ STAT memory 225. An example of information #2 is packet count information stored in the embodiment of Figure 10 in PFQ STAT memory 225. These two types of information are, in one embodiment of Figure 48, stored in different external memory devices.

**Page 60, line 32 (the header), should be replaced with:**

BACKPRESSURING USING SERIAL BUS:

**Page 70, the first full paragraph, should be replaced with:**

There is at most one segmentation process going on per port. Incoming burst data for a port is accumulated into a corresponding one of the 64-byte buffers in data SRAM 908. The maximum amount of data that can be stored in the 64-byte buffer (either 48, 56, or 64 bytes) is determined by the application type of the flow. The 64-byte format is used for packet data. The 48-byte format is used for AAL5 ATM data. The 56-byte format is used for non-AAL5 ATM data. Control information for the data in a 64-byte

buffer is queued into a location in queue FIFO 912 that corresponds to the particular 64-byte buffer. Segmentation table and statistics information is written back to the segmentation table and statistics memory 907. Input phase state machine 904 contains a six-bit data byte counter (not shown) that increments the six-bit "cell length" (also called the "chunk length") count value (stored in segmentation table memory 907) upon receiving each incoming data byte for a 64-byte buffer. When the number of bytes in the 64-byte buffer reaches the maximum amount allowed by the format used, then the data in the 64-byte buffer is ready for transfer to memory manager block 204.

**The paragraph from page 74, line 30 through page 75, line 8, should be replaced with:**

Although in this embodiment external memory 217 is ZBT (zero bus turnaround) SRAM, external memory 217 may in other embodiments be another type of memory such as, for example, DRAM. In some embodiments, bandwidth to external memory 217 is increased by realizing external memory 217 in multiple integrated circuit memory devices, where each is accessed via a different one of a plurality of interface ports controlled by the same memory controller block 1007. Memory controller 1007 in Figure 58 is programmable to interface with pipelined memory. In one such embodiment, the four memory locations where a chunk of data is stored are read in sequence. Although the reading of the first of the four locations requires multiple clock cycles, pipelining is employed such that the other of the four locations are read, one location on each subsequent clock cycle.

**The paragraph from page 84, line 20 through page 85, line 23, should be replaced with:**

Enqueue data pipe block 1115 converts data from 128-bit wide to 64-bit wide for entry into data SRAM 1107. Enqueue data pipe block 1115 also provides the appropriate delay necessary for enqueue state machine block 1101 to obtain the pointer needed to store the chunk into data SRAM 1107. Dequeue data pipe block 1116 delays data to the output FIFO block 1110 so that the header can be inserted and the CRC checked for L2 packets before the last word is sent. Header SRAM block 1106 is a

256x64 bit internal dual port SRAM that stores header information on a per-port basis. The memory is organized as 128 sixteen-byte buffers so that two sixteen-byte headers are stored for each output port. Output FIFO block 1110 is a 32 x 88 bit FIFO implemented using dual port memory. The output FIFO is large enough to store three maximum size chunks. The output FIFO has two full indications (Almost Full and Full) and two empty indications (Almost Empty and Empty). Almost Full is asserted when the FIFO has ten locations left. Full is asserted when the FIFO has nine or fewer locations left. By indicating that the FIFO is almost full when it has only ten locations left, the dequeue state machine block 1108 can then send one more complete chunk. When the FIFO is one word beyond Almost Full, then the Full signal is asserted thereby indicating that once the current chunk is completed that no more chunks are to be transferred until Full goes inactive. Almost Empty is asserted when one word is left in the output FIFO. Empty is asserted when the output FIFO is empty. Outgoing SPI interface block 206 ORs the two empty signals together to determine when to pop the output FIFO. Outgoing SPI interface block 206 also uses the two empty signals to determine when only one word is left in the FIFO. This is necessary because the minimum POP size is two clock cycles, but when only one word is left the FIFO only has valid data for the first read of the FIFO. If Empty is active but Almost Empty is inactive, then only one word is in the FIFO. Dequeue state machine block 1108 causes data to be loaded into output FIFO block 1110 as long as output port calendar block 1109 indicates chunks are available and output FIFO block 1110 is not full. If output FIFO block 1110 goes full, then the current chunk being transferred is completed before reassembly block 1100 stops sending data to output FIFO block 1110. Outgoing SPI interface block 206 pops data from the output FIFO block 1110 as long as FIFO block 1110 is not empty and outgoing SPI interface block 206 is not inserting control words into the data stream. FIFO block 1110 allows outgoing SPI interface block 206 to control the flow of data from reassembly block 1100 so it has time to add the control words into the output data stream (coming out of outgoing SPI interface block 206) between bursts.

IN THE CLAIMS:

Please amend the claims as follows.

Cancel Claims 1-22 and 25-44, without prejudice.

Add new Claims 45-59 as follows.

45.(New) A multi-service segmentation and reassembly (MS-SAR) integrated circuit employing a plurality of logical input ports, the MS-SAR integrated circuit comprising:

bus interface circuitry; and

a segmentation engine that receives a plurality of flows via the bus interface circuitry, each of the plurality of flows comprising a plurality of packets, the segmentation engine segmenting the packets of the flows on a per logical input port basis such that at any one time at most substantially one packet is being segmented for each of the plurality of logical input ports, the segmenting of a packet resulting in a plurality of segments.

46.(New) The MS-SAR integrated circuit of Claim 45, wherein the segmentation engine receives a packet as a plurality of bursts, and wherein the segmentation engine accumulates several bursts to form a segment.

47.(New) The MS-SAR integrated circuit of Claim 45, further comprising:

a memory manager, wherein the segmentation engine passes a plurality of chunks to the memory manager, each of the chunks comprising one of the segments, the memory manager storing each of the chunks into a corresponding memory buffer.

48.(New) The MS-SAR integrated circuit of Claim 45, wherein the segmentation engine segments one and only one packet for each of the plurality of logical input ports.

49.(New) The MS-SAR integrated circuit of Claim 45, wherein the logical input ports of are active logical input ports, and wherein in addition to the active logical input ports there are inactive logical input ports.

50.(New) The MS-SAR integrated circuit of Claim 45, wherein the segmentation engine maintains at most one segmentation context per packet being segmented.

51.(New) The MS-SAR integrated circuit of Claim 50, wherein each segmentation context comprises a byte count value and a cyclic redundancy check value.

52.(New) The MS-SAR integrated circuit of Claim 45, wherein the plurality of flows pass from the bus interface circuitry, through a lookup engine, and to the segmentation engine.

53.(New) The MS-SAR integrated circuit of Claim 45, wherein the segmentation engine also receives a flow of cells via the bus interface circuitry, the segmentation engine segmenting the cells into segments.

54.(New) A multi-service segmentation and reassembly (MS-SAR) integrated circuit employing a plurality of logical input ports, the MS-SAR integrated circuit comprising:

bus interface circuitry; and

means for receiving a plurality of flows via the bus interface circuitry, each of the plurality of flows comprising a plurality of packets, the means being for segmenting the packets of the flows on a per logical input port basis such that at any one time at most substantially one packet is being segmented for each of the plurality of logical input ports, the segmenting of a packet resulting in a plurality of segments.

55.(New) A method, comprising:

receiving a plurality of flows of network information, the plurality of flows being received on a plurality of active logical input ports of an integrated circuit, some of the flows being flows of packets, others of the flows being flows of cells;

performing a segmentation process on each of the flows such that there is no more than approximately one segmentation process per active logical input port going on at any one time on the integrated circuit, each segmentation process generating one or more segments of the network information;

storing the segments of the network information into a corresponding set of memory buffers; and

retrieving the segments from the memory buffers and outputting the segments such that the network information is output from the integrated circuit.

56.(New) The method of Claim 55, wherein at most substantially one segmentation context is maintained for each of the active logical input ports.

57.(New) The method of Claim 55, wherein the integrated circuit performs an ingress function, the network information passing from the integrated circuit and to a switch fabric.

58.(New) The method of Claim 55, wherein the integrated circuit performs an egress function, the network information passing from a fiber optic cable and to the integrated circuit.

59.(New) A method, comprising:

receiving a plurality of flows of network information onto a plurality of active logical input ports, some of the flows being flows of packets, others of the flows being flows of cells;

step for performing segmentation on the plurality of flows of network information such that no more than approximately one segmentation context is maintained for each of the active logical input ports, the segmentation of the plurality of flows resulting in a plurality of segments;

storing the segments in a memory; and

retrieving the segments from the memory and supplying the network information to a switch fabric.

## REMARKS

Consideration and allowance is respectfully requested. Before examination, please make the amendments as set forth above. After entry of this Preliminary Amendment, Claims 23, 24 and 45-59 are pending. If the Examiner would like to discuss any aspect of this application, the Examiner is requested to contact the undersigned at (925) 485-9923.

I hereby certify that this correspondence is being deposited with the United States Postal Service as "Express Mail Post Office to Addressee" addressed to Box Patent Application, Assistant Commissioner for Patents, Washington, D.C. 20231, on October 12, 2001, as

Express Mail No.: EL928548091US.

T. Lester Wallace

Lester Wallace — Signature

Oct. 12, 2001 — Date of Signature

Respectfully submitted,

Lester Wallace

T. Lester Wallace
Attorney for Applicants
Reg. No. 34,748

## VERSION WITH MARKINGS TO SHOW CHANGES MADE

THE DRAWINGS:
A copy of Figure 52 with the corrections indicated in red is attached.

THE SPECIFICATION:

**Page 1, before the first sentence of the specification, please add the following header and paragraph:**

CROSS-REFERENCE TO RELATED APPLICATION

This application is a divisional of application serial number 09/851,565, filed May 8, 2001.

**The paragraph from page 14, line 31 through page 15, line 10:**

Figure 4 is a simplified diagram of a router 100 in accordance with an embodiment of the present invention. Router 100 includes a plurality of line cards 101-104, a switch fabric 105 and a central processing unit (CPU) 106. The line cards 101-104 are coupled to switch fabric 105 by parallel buses 107-114. In the present example, each of parallel buses 107-114 is a 16-bit SPI-4, Phase II, LVDS parallel bus operating at 400 MHz at a double data rate (DDR). CPU 106 is coupled to line cards 101-104 by another parallel bus 131. In the present example, parallel bus **[**130**]** **131** is a 32-bit PCI bus. In this example, each of the line cards can receive network communications in multiple formats. For example, line card 101 is coupled to a fiber optic cable 115 such that line card 101 can receive from cable 115 network communications at OC-192 rates in packets, ATM cells, and/or AAL5 cells. AAL5 cells are considered a type of ATM cell.

**Page 33, the first full paragraph:**

In accordance with egress application type 11, segmentation block 203 does not **perform** segmenting per se but rather forwards the 64-byte chunk to memory manager block 204 via the per-port FIFO mechanism described above. Memory manager block 204 issues an enqueue command via enqueue command line 320, and stores the 64-byte chunk in payload memory 217. Per flow queue block 207 adds the BID to the per flow queue for flow #1. The same process flow occurs for the second and third switch cells associated with flow #1.

**Page 34, the second full paragraph:**

Figure 32 illustrates the general flow of information out of egress MS-SAR 200 in one scenario when the linked lists for flow #1 and flow #2 are dequeued. As each BID is dequeued from a per flow queue, the associated 64-byte chunk is read from payload memory 217 and is supplied via [128-bite] **128-bit** wide data bus 324 to reassembly block 205. As explained above in connection with the ingress mode, reassembly block 205 maintains one reassembly queue for each of the 64 logical output ports. If two flows share the same logical output port, then the entire linked list for one flow must be dequeued before the linked list for the next flow having the same logical output port is dequeued. In the example of Figure 8, flow #1 and flow #2 have different logical output ports. The dequeue command as received by memory manager block 204 from per flow queue block 207 contains the port ID (PID) of the one of the 64 logical output ports of line card 103.

**Page 36, the second full paragraph:**

The AAL5 trailer also contains **a** sixteen-bit "length of payload" field that indicates the length of the payload of the packet being reassembled. Reassembly block 205 maintains a partial packet byte count value for the packet. As each 64-byte chunk of the packet is received from memory manager block 204, reassembly block 205 adds the number of bytes received to the partial packet byte count value. After the last 64-byte chunk for the packet has been received and the partial packet byte count value has been added to, reassembly block 205 compares the now complete packet byte count value with the "length of payload" value from the AAL5 trailer. The two should match.

**Page 59, the fourth full paragraph:**

Figure 52 is a diagram of an external memory device 529 that is coupled to a control integrated circuit. External memory device 529 may, for example, be external memory 225 of Figure 10. External memory device 529 stores two types of information, information #1 and information #2, both of which must be accessed within a particular amount of time related to the rate of incoming information. Where the control integrated circuit is clocked by a clock signal, this particular amount of time can be referred to as

[eight] **four** clock periods. In the example to the left of Figure 52, each piece of information can be accessed in two clock periods. Both pieces of information are stored in the same external memory device requiring one to be accessed before the other. A total of four clock periods is therefore required to access both pieces of information.

**Page 60, the second full paragraph:**

In accordance with one novel aspect, two external memories are used. Information #1 is stored in the first external memory 530 and information #2 is stored in second external memory 531. The two external memories are accessed at the same time in parallel. If external memories 530 and 531 are the same type of external memory 529, then these external memories have access times of two clock periods and both information #1 and information #2 are accessed within the required two clock periods. It is to be understood, of course, that the [eight] **four** and two in the example are used only to illustrate the technique of accessing memories in parallel to facilitate handling higher data throughput rates. The technique here is not limited to the numbers in this example. An example of information #1 is cell count information stored in the embodiment of Figure 10 in PFQ STAT memory 225. An example of information #2 is packet count information stored in the embodiment of Figure 10 in PFQ STAT memory 225. These two types of information are, in one embodiment of Figure 48, stored in different external memory devices.

**Page 60, line 32 (the header):**

[BACKPRESSING] **BACKPRESSURING** USING SERIAL BUS:

**Page 70, the first full paragraph:**

There is at most one segmentation process going on per port. Incoming burst data for a port is accumulated into a corresponding one of the 64-byte buffers in data SRAM 908. The maximum amount of data that can be stored in the 64-byte [bluffer] **buffer** (either 48, 56, or 64 bytes) is determined by the application type of the flow. The 64-byte format is used for packet data. The 48-byte format is used for **AAL5** ATM data. The 56-byte format is used for [other] **non-AAL5** ATM data. Control information for the data in a 64-byte buffer is queued into a location in queue FIFO 912 that corresponds to

the particular 64-byte buffer. Segmentation table and statistics information is written back to the segmentation table and statistics memory 907. Input phase state machine 904 contains a six-bit data byte counter (not shown) that increments the six-bit "cell length" (also called the "chunk length") count value (stored in segmentation table memory 907) upon receiving each incoming data byte for a 64-byte buffer. When the number of bytes in the 64-byte buffer reaches the maximum amount allowed by the format used, then the data in the 64-byte buffer is ready for transfer to memory manager block 204.

**The paragraph from page 74, line 30 through page 75, line 8:**

Although in this embodiment external memory 217 is ZBT (zero bus turnaround) SRAM, external memory 217 may in other embodiments be another type of memory such as, for example, DRAM. In some embodiments, bandwidth to external memory 217 is increased by realizing external memory 217 in multiple integrated circuit memory devices, where each is accessed via a different one of a plurality of interface ports controlled by the same memory controller block 1007. Memory controller 1007 in Figure 58 is programmable to interface with pipelined memory. In one such embodiment, the four memory locations where a chunk of data is stored are read in sequence. Although the reading of the first of the four locations requires multiple clock cycles, [piplelining] **pipelining** is employed such that the other of the four locations are read, one location on each subsequent clock cycle.

**The paragraph from page 84, line 20 through page 85, line 23:**

Enqueue data pipe block 1115 converts data from 128-bit wide to 64-bit wide for entry into data SRAM 1107. Enqueue data pipe block 1115 also provides the appropriate delay necessary for enqueue state machine block 1101 to obtain the pointer needed to store the chunk into data SRAM 1107. Dequeue data pipe block 1116 delays data to the output FIFO block 1110 so that the header can be inserted and the CRC checked for L2 packets before the last word is sent. Header SRAM block 1106 is a 256x64 bit internal dual port SRAM that stores header information on a per-port basis. The memory is organized as 128 sixteen-byte buffers so that two sixteen-byte headers are stored for each output port. Output FIFO block 1110 is a 32 x 88 bit FIFO

implemented using dual port memory. **The** [Th e] output FIFO is large enough to store three maximum size chunks. The output FIFO has two full indications (Almost Full and Full) and two empty indications (Almost Empty and Empty). Almost Full is asserted when the FIFO has ten locations left. Full is asserted when the FIFO has nine or fewer locations left. By indicating that the FIFO is almost full when it has only ten locations left, the dequeue state machine block 1108 can then send one more complete chunk. When the FIFO is one word beyond Almost Full, then the Full signal is asserted thereby indicating that once the current chunk is completed that no more chunks are to be transferred until Full goes inactive. Almost Empty is asserted when one word is left in the output FIFO. Empty is asserted when the output FIFO is empty. Outgoing SPI interface block 206 ORs the two empty signals together to determine when to pop the output FIFO. Outgoing SPI interface block 206 also uses the two empty signals to determine when only one word is left in the FIFO. This is necessary because the minimum POP size is two clock cycles, but when only one word is left the FIFO only has valid data for the first read of the FIFO. If Empty is active but Almost Empty is inactive, then only one word is in the FIFO. Dequeue state machine block 1108 causes data to be loaded into output FIFO block 1110 as long as output port calendar block 1109 indicates chunks are available and output FIFO block 1110 is not full. If output FIFO block 1110 goes full, then the current chunk being transferred is completed before reassembly block 1100 stops sending data to output FIFO block 1110. Outgoing SPI interface block 206 pops data from the output FIFO block 1110 as long as FIFO block 1110 is not empty and outgoing SPI interface block 206 is not inserting control words into the data stream. FIFO block 1110 allows outgoing SPI interface block 206 to control the flow of data from reassembly block 1100 so it has time to add the control words into the output data stream (coming out of outgoing SPI interface block 206) between bursts.

THE CLAIMS:

Cancel Claims 1-22 and 25-44.

Add new Claims 45-59 as follows.

45.(New) A multi-service segmentation and reassembly (MS-SAR) integrated circuit employing a plurality of logical input ports, the MS-SAR integrated circuit comprising:

bus interface circuitry; and

a segmentation engine that receives a plurality of flows via the bus interface circuitry, each of the plurality of flows comprising a plurality of packets, the segmentation engine segmenting the packets of the flows on a per logical input port basis such that at any one time at most substantially one packet is being segmented for each of the plurality of logical input ports, the segmenting of a packet resulting in a plurality of segments.

46.(New) The MS-SAR integrated circuit of Claim 45, wherein the segmentation engine receives a packet as a plurality of bursts, and wherein the segmentation engine accumulates several bursts to form a segment.

47.(New) The MS-SAR integrated circuit of Claim 45, further comprising:

a memory manager, wherein the segmentation engine passes a plurality of chunks to the memory manager, each of the chunks comprising one of the segments, the memory manager storing each of the chunks into a corresponding memory buffer.

48.(New) The MS-SAR integrated circuit of Claim 45, wherein the segmentation engine segments one and only one packet for each of the plurality of logical input ports.

49.(New) The MS-SAR integrated circuit of Claim 45, wherein the logical input ports of are active logical input ports, and wherein in addition to the active logical input ports there are inactive logical input ports.

50.(New) The MS-SAR integrated circuit of Claim 45, wherein the segmentation engine maintains at most one segmentation context per packet being segmented.

51.(New) The MS-SAR integrated circuit of Claim 50, wherein each segmentation context comprises a byte count value and a cyclic redundancy check value.

52.(New) The MS-SAR integrated circuit of Claim 45, wherein the plurality of flows pass from the bus interface circuitry, through a lookup engine, and to the segmentation engine.

53.(New) The MS-SAR integrated circuit of Claim 45, wherein the segmentation engine also receives a flow of cells via the bus interface circuitry, the segmentation engine segmenting the cells into segments.

54.(New) A multi-service segmentation and reassembly (MS-SAR) integrated circuit employing a plurality of logical input ports, the MS-SAR integrated circuit comprising:

bus interface circuitry; and

means for receiving a plurality of flows via the bus interface circuitry, each of the plurality of flows comprising a plurality of packets, the means being for segmenting the packets of the flows on a per logical input port basis such that at any one time at most substantially one packet is being segmented for each of the plurality of logical input ports, the segmenting of a packet resulting in a plurality of segments.

55.(New) A method, comprising:

receiving a plurality of flows of network information, the plurality of flows being received on a plurality of active logical input ports of an integrated circuit, some of the flows being flows of packets, others of the flows being flows of cells;

performing a segmentation process on each of the flows such that there is no more than approximately one segmentation process per active logical input port going on at any one time on the integrated circuit, each segmentation process generating one or more segments of the network information;

storing the segments of the network information into a corresponding set of memory buffers; and

retrieving the segments from the memory buffers and outputting the segments such that the network information is output from the integrated circuit.

56.(New) The method of Claim 55, wherein at most substantially one segmentation context is maintained for each of the active logical input ports.

57.(New) The method of Claim 55, wherein the integrated circuit performs an ingress function, the network information passing from the integrated circuit and to a switch fabric.

58.(New) The method of Claim 55, wherein the integrated circuit performs an egress function, the network information passing from a fiber optic cable and to the integrated circuit.

59.(New) A method, comprising:

receiving a plurality of flows of network information onto a plurality of active logical input ports, some of the flows being flows of packets, others of the flows being flows of cells;

step for performing segmentation on the plurality of flows of network information such that no more than approximately one segmentation context is maintained for each of the active logical input ports, the segmentation of the plurality of flows resulting in a plurality of segments;

storing the segments in a memory; and

retrieving the segments from the memory and supplying the network information to a switch fabric.

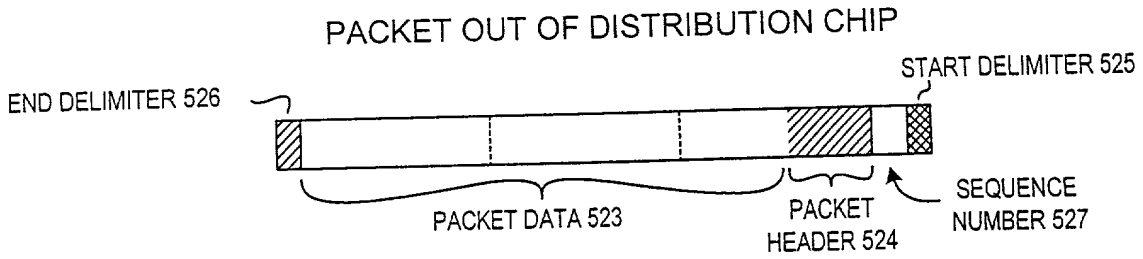
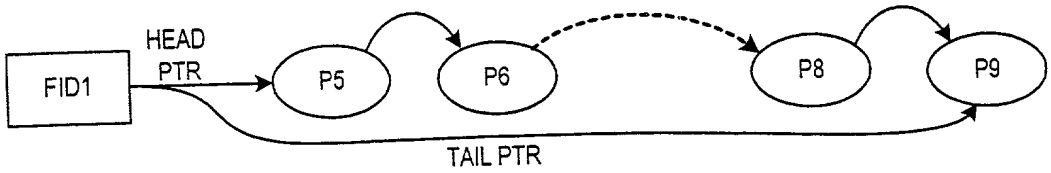
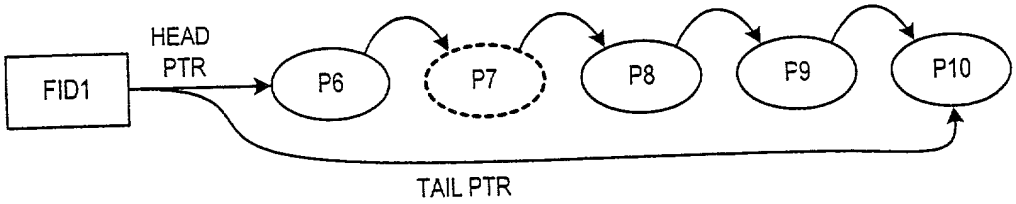



FIG. 49




PACKET QUEUE

FIG. 50




PACKET QUEUE

FIG. 51

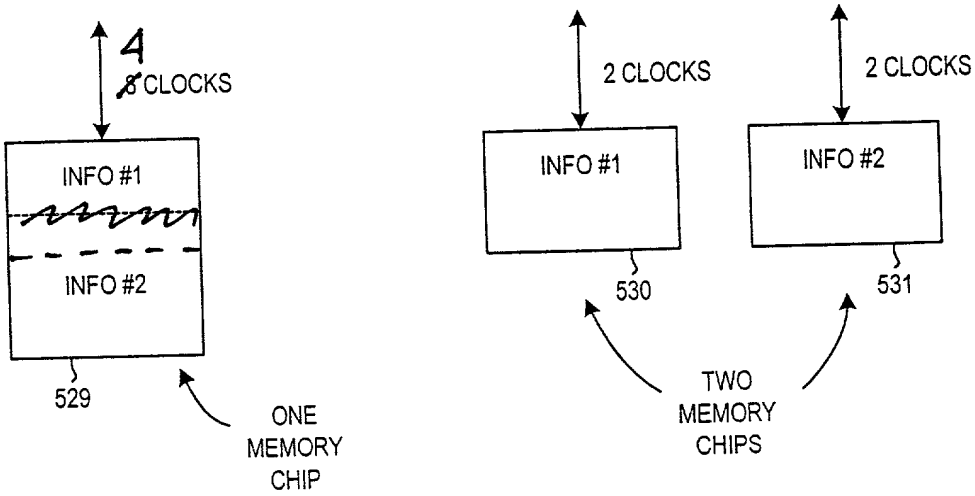



FIG. 52